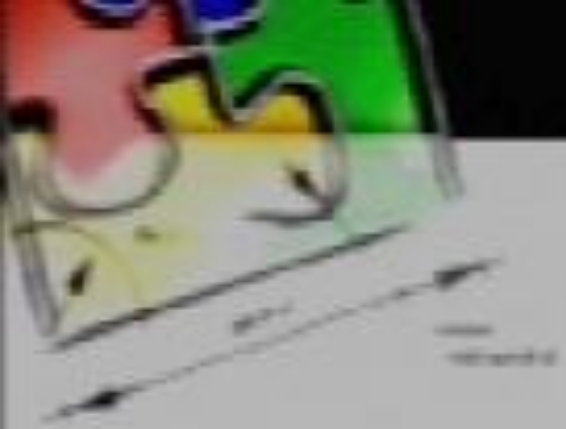


Ordering and Making Pizza for 120 Million

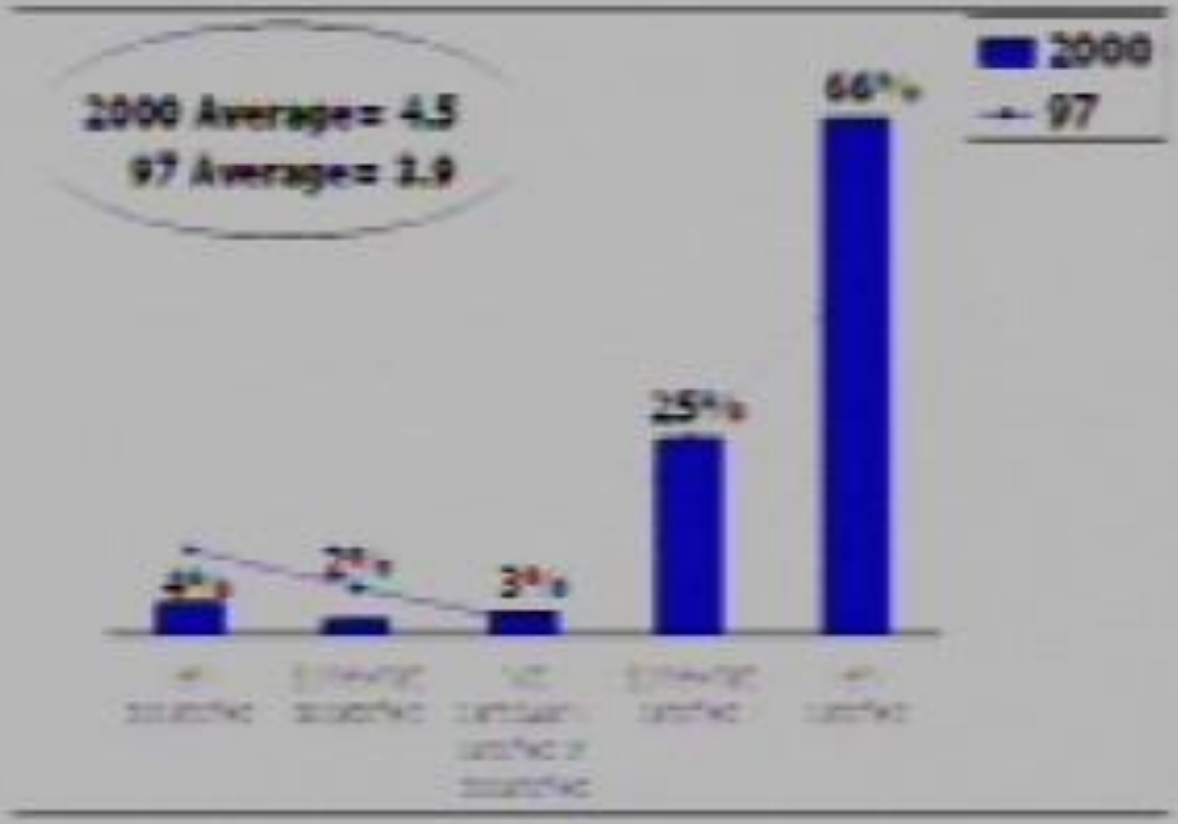
Designing and Building the Next Office

Agenda

- Designing a New Office
  - Office customer challenges
  - Understanding our customers
  - Using customer feedback understanding in design
- Building Office
  - How we're organized
  - The development cycle
  - The code
  - The life of an Office developer
  - What we do well
  - What we don't do so well

Challenges – The Pizza Problem…

- 120 million relatively happy and loyal customers
  - Why should I get the next version?



Highlights

- 2/3 of users give Office 2000 a rating of 5 out of 5
- 91% are at least somewhat satisfied with Office 2000

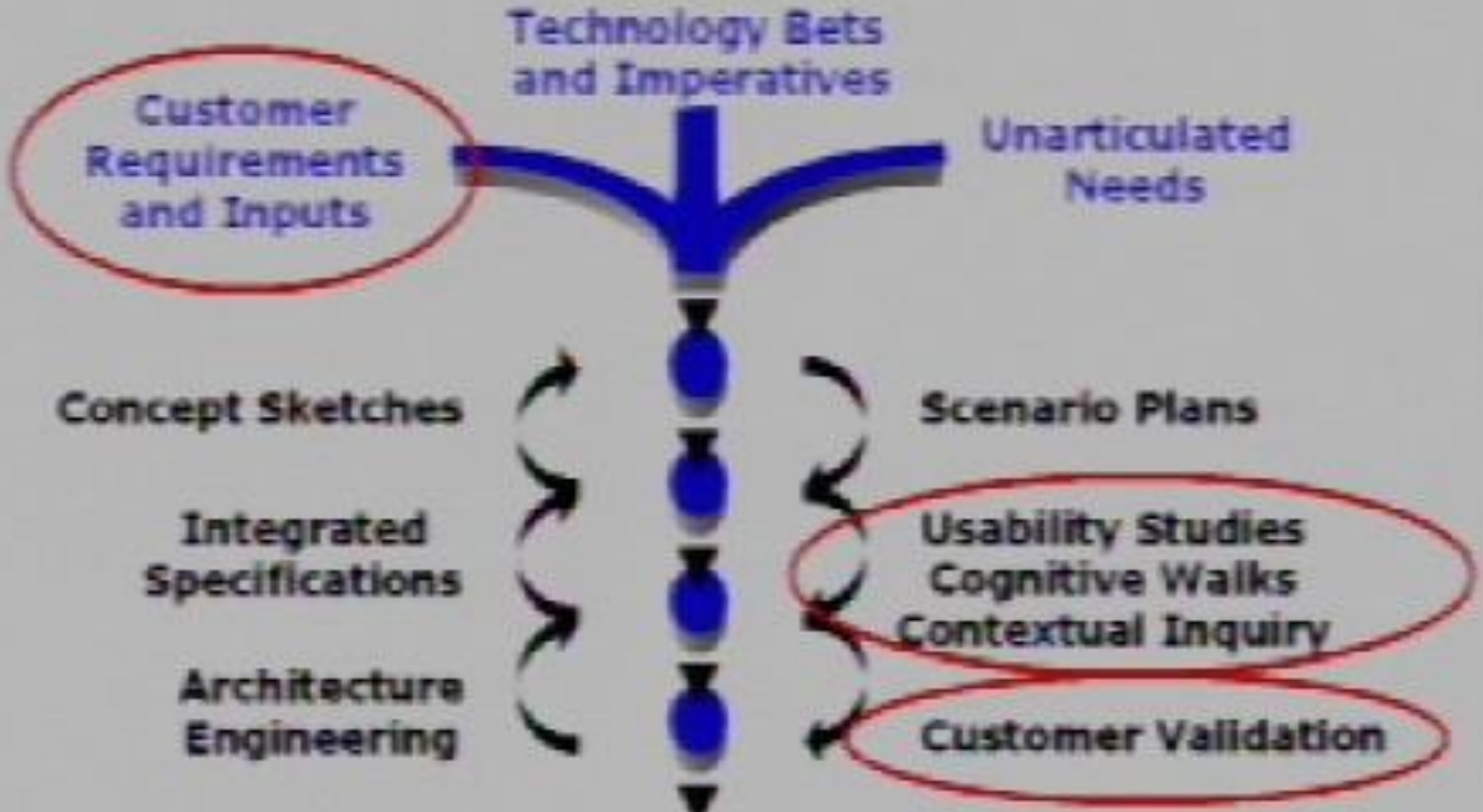
Challenges – The Pizza Problem…

- Wide spectrum of target customers
  - From end-users to administrators
- Want new features but no new learning
  - How do I migrate the users and the current learning
- Must be innovative but fully compatible
  - New features but no surprises (file formats, macros, solutions)
- Designed to be flexible but lockable
  - Fully customizable and configurable – but lockable if I so wish
- Fully loaded but not bloated
  - My 20% of the features are most important.
  - The other 80% should not get in my way

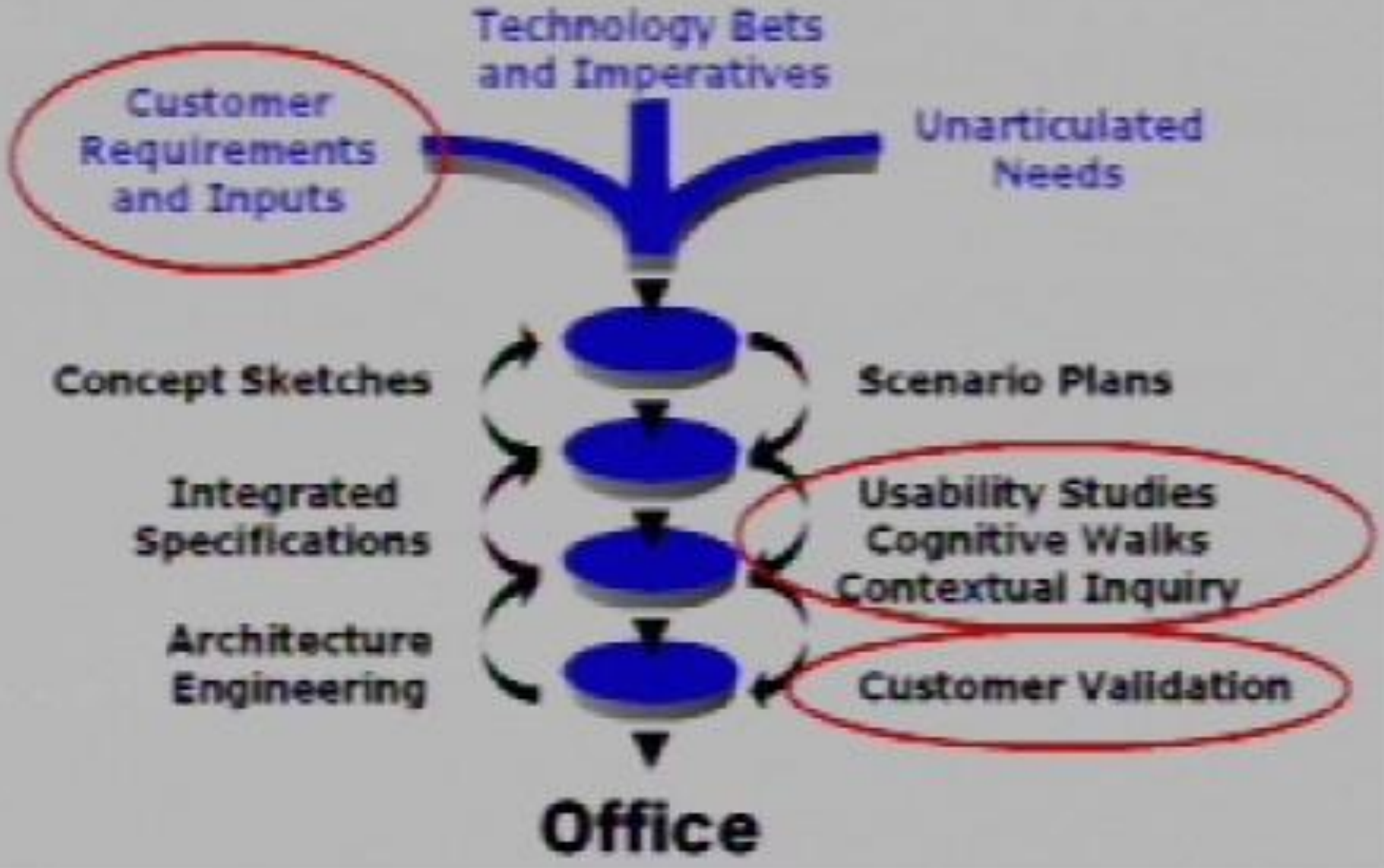
Understanding the customer better

- Instrumented versions
  - Log everything done by users
  - Some "duhs", some surprises
- Usability studies
  - Get to see and feel the customer's pain
- Customer visits
  - Everyone does them
  - Watch users work
- Advisory Council
  - From 15 hand-picked companies
  - Meet in Redmond before and during dev cycle

Designing for innovation



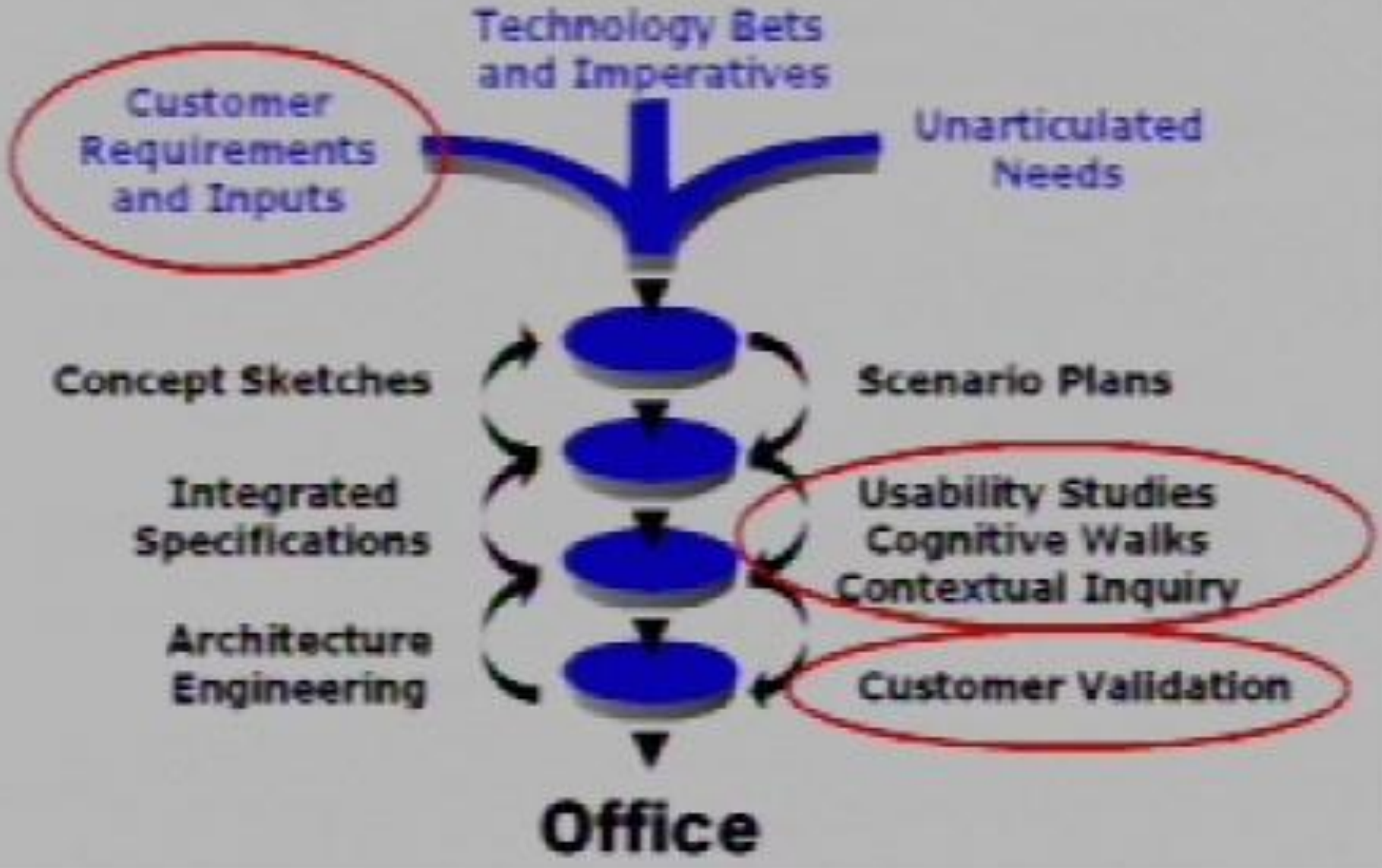
Designing for innovation



How We're Organized

- Roles:
  - Core:
    - Dev, Test, Prog. Mgmt., User Ed., Marketing
  - Others:
    - Product Planners
    - Designers
    - Usability Specialists
    - PSS
    - Localizers

Designing for innovation



How We're Organized

- Roles:
  - Core:
    - Dev, Test, Prog. Mgmt., User Ed., Marketing
  - Others:
    - Product Planners
    - Designers
    - Usability Specialists
    - PSS
    - Localizers

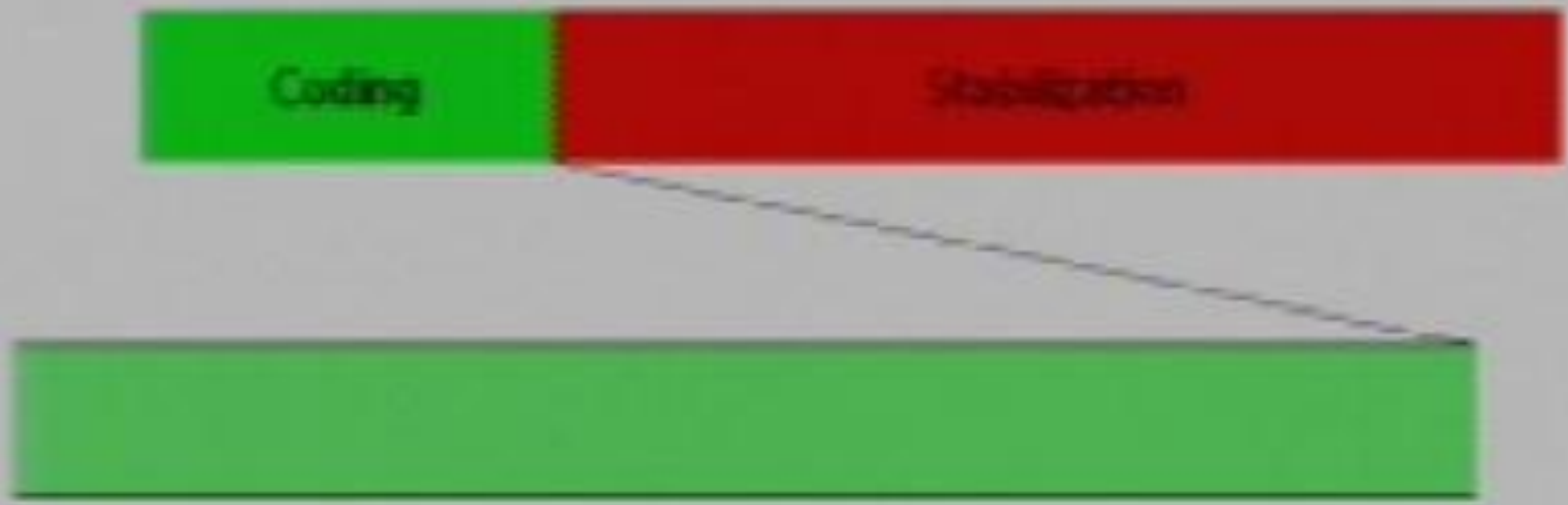
How We're Organized

- Groups
  - Applications Teams
    - Full dev, test, pm teams
    - Develop app-specific features/code
  - Office Shared Feature Teams
    - Also full dev, test, pm teams
    - Develop shared features
    - Responsible for shared code + integration
  - Office "services"
    - Marketing, UE, Prod. Planning, Localization, etc.
  - MOSE
    - Develop QFEs and SRs

How We're Organized

Development Timeline

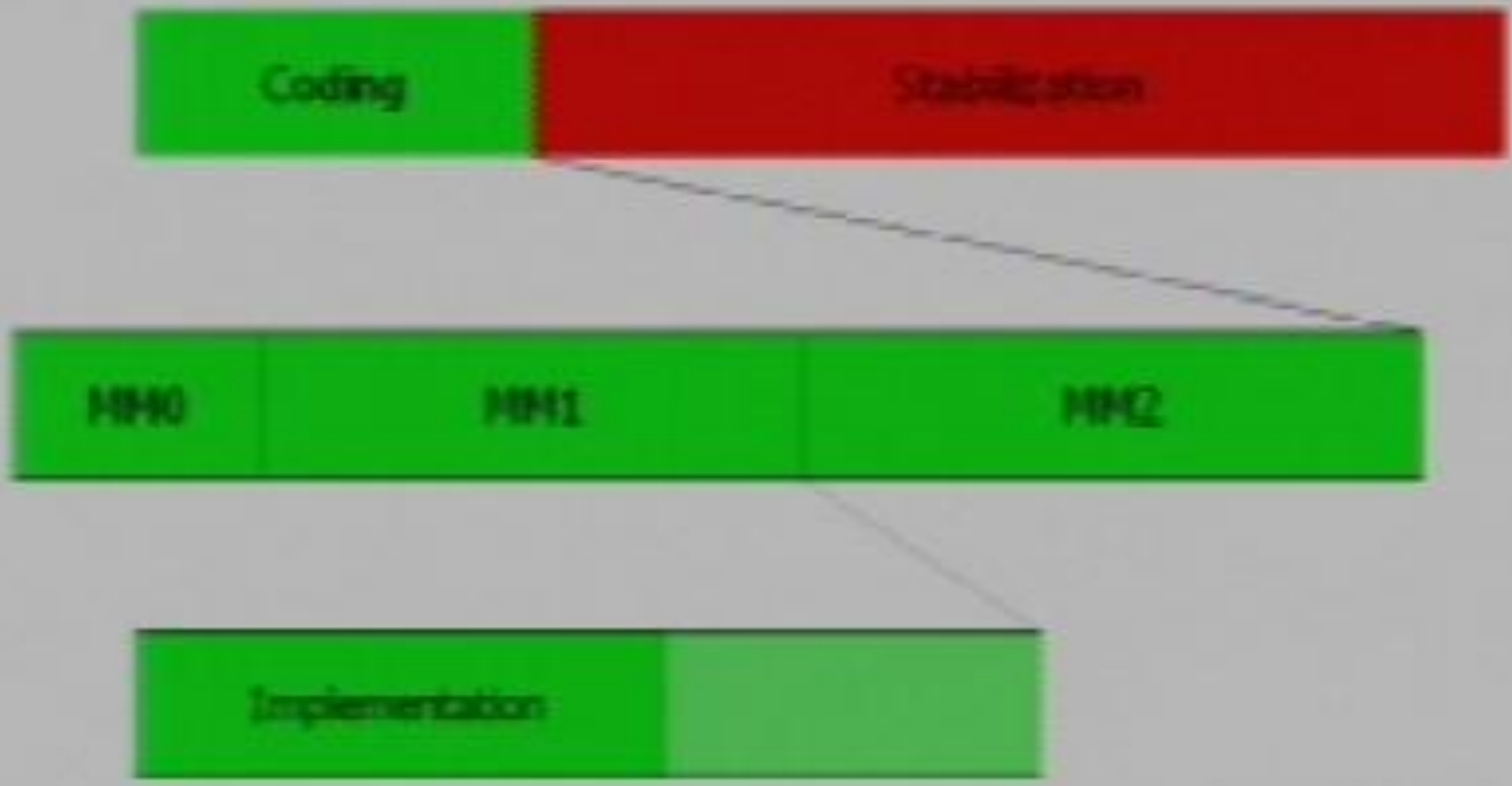
Development Timeline



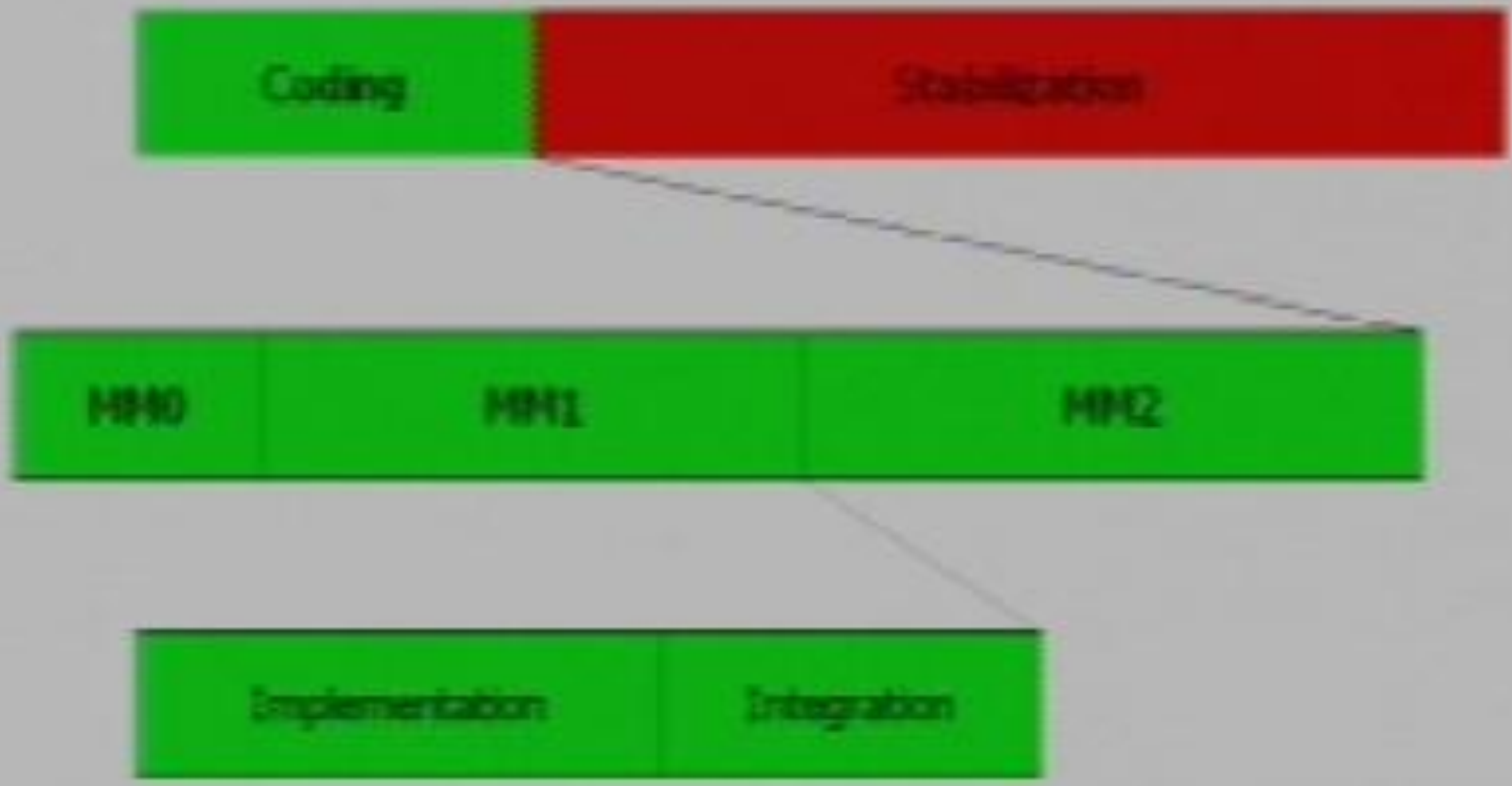
Development Timeline

Development Timeline

Development Timeline



Development Timeline



Development Timeline

Development Timeline

Code
Complete

Development Timeline

Development Timeline

Code Complete

Beta 1

TRP

Development Timeline

Development Timeline

Development Timeline

Code Complete

Development Timeline

Code Complete

Beta 1

CBF

Beta 2

RTM!

The life of an Office developer

MMO

- Features are assigned

The life of an Office developer

MMO

- Features are assigned
- Designs are done
- Design reviews
- Features are scheduled
- Schedule is rolled up and balanced

The life of an Office developer

The life of an Office developer

MMI-MMn

- Implementation starts
- Different for Office shared developers v. App developers
- Typical day:
  - Write code / fix bugs
    - Incremental builds during day
  - Maybe check-in
    - Sync to checkpoint
    - Run all CITs
    - Profile perf analysis
  - Full OHOME build overnight

The life of an Office developer

The life of an Office developer

Debug

- Life centered around the bug list
- Weekly goals
  - Total count
  - Bug age
  - Priority (eg beta bugs)
- Group triage kicks in at some point
- OBTriage kicks in a little later

About the Code

- All apps written in C
  - Excel, Word
- ...or in C++
  - MSO, Access, Powerpoint, Outlook

About the Code

How much code is there?

About the Code

How much code is there?

Office 2000	
Office	2,178,703
Word	2,021,502
Excel	1,535,233
Outlook	1,498,153
Access	1,179,398
PowerPoint	790,403
Total	**7,705,239**

About the Code

How much code is there?

Office 2000	
Office	2,178,703
Word	2,021,502
Excel	1,535,233
Outlook	1,498,153
Access	1,179,398
PowerPoint	790,403
Total	**7,705,239**

Office 97	
Office	725,329
Word	1,176,526
Excel	1,371,296
Outlook	645,956
Access	362,988
PowerPoint	420,131
Total	**4,056,270**

About the Code

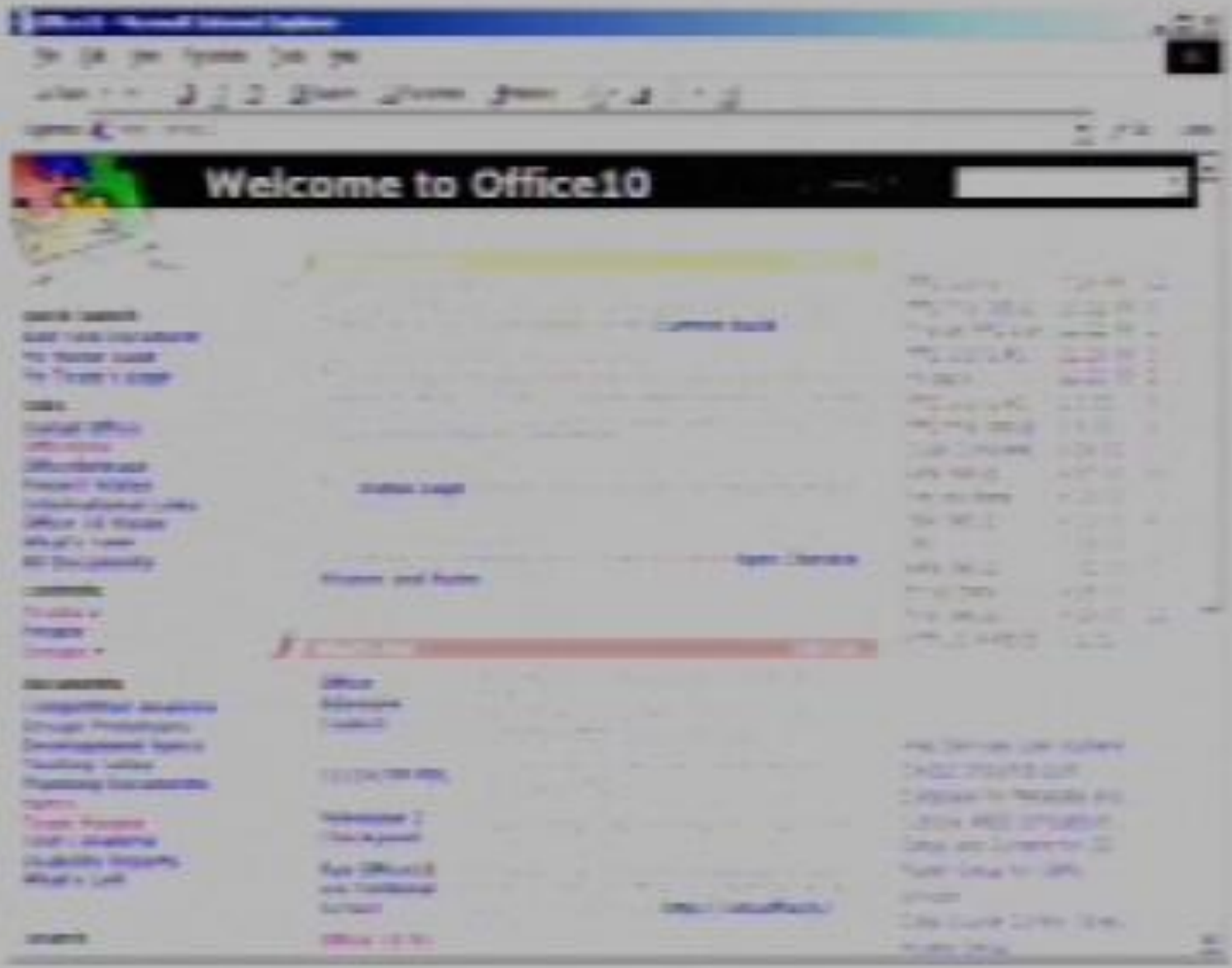
- Word in Office 2000
  - 24,627 check-ins
  - 145 developers making changes
  - Touched 932,428 lines of code
  - 80% of changes by 21% of developers
  - LOCs grew by 71.8%

What we do well

(for all disciplines)

- Product Vision
- Setup from day one
- Teams organized by feature areas, not apps
  - Where applicable possible
- Unified test harness
  - Execution, Results reporting
- Rigorous spec inspections
- Knowledge Management
  - Single bug database (plus FastRaid)
  - Single internal site for project info
  - Single schedule db for all apps

http://office10

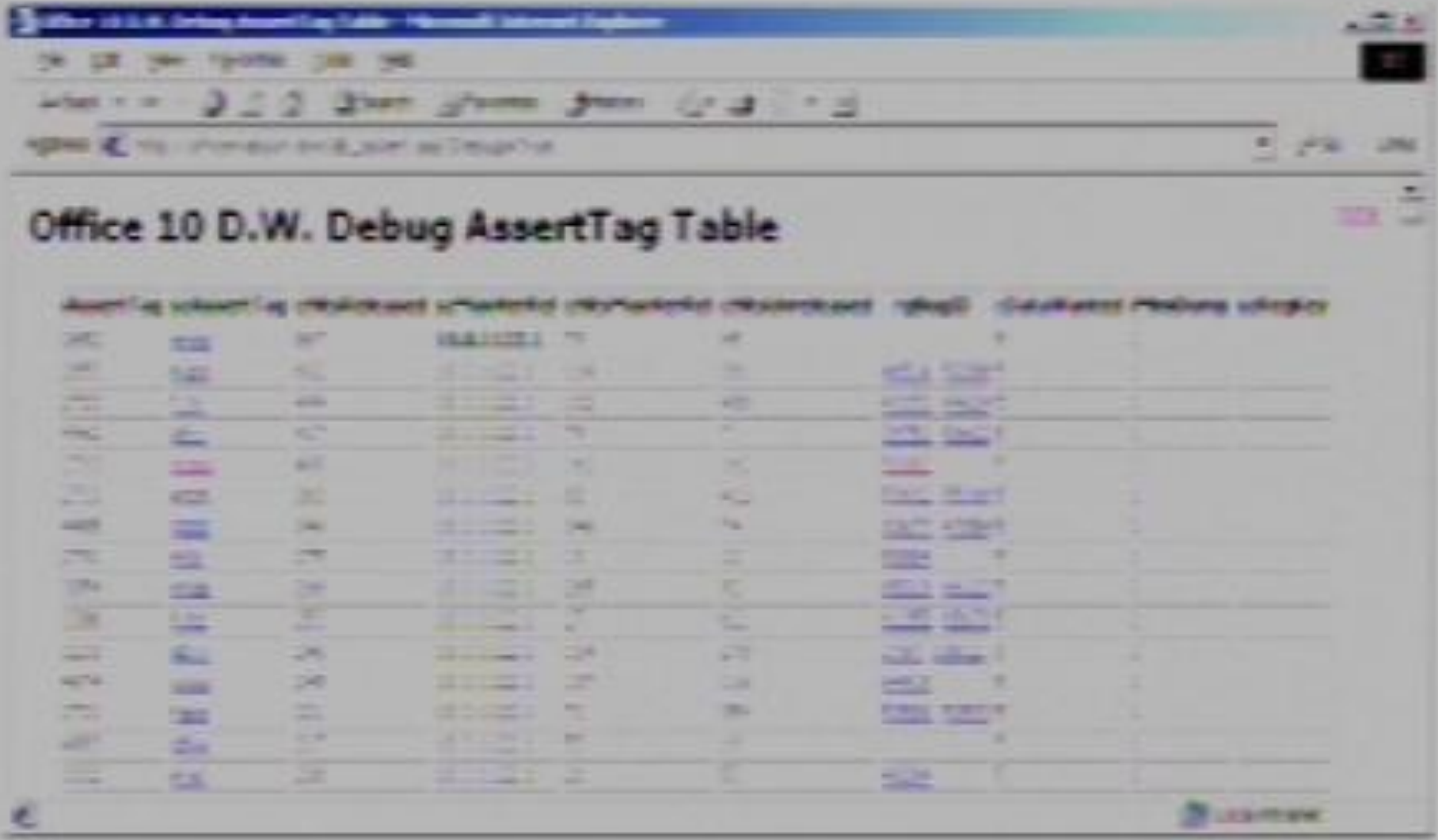


What we do well

(for developers)

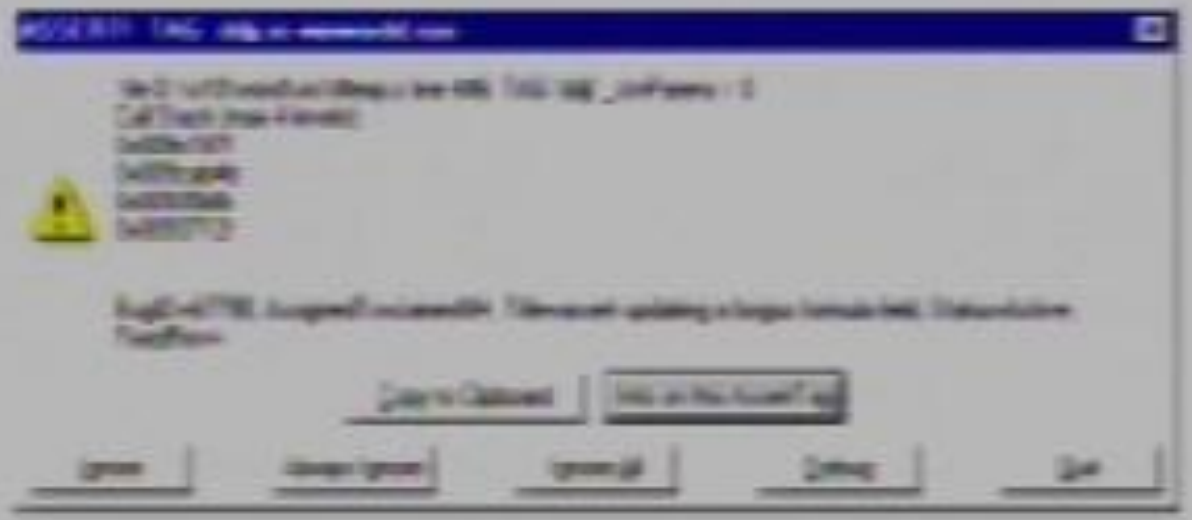
- Single unified build system
  - Easy for anyone to build anything
  - One tool makes it easy to enable new things
  - Even pre-installed images for new build machines
- Regular complete builds
  - Checkpoints
  - Large no. of tests
  - Including international builds, etc.
- Logging relevant data
  - Build problems, performance
  - Asserts, GPFs

AssertTags

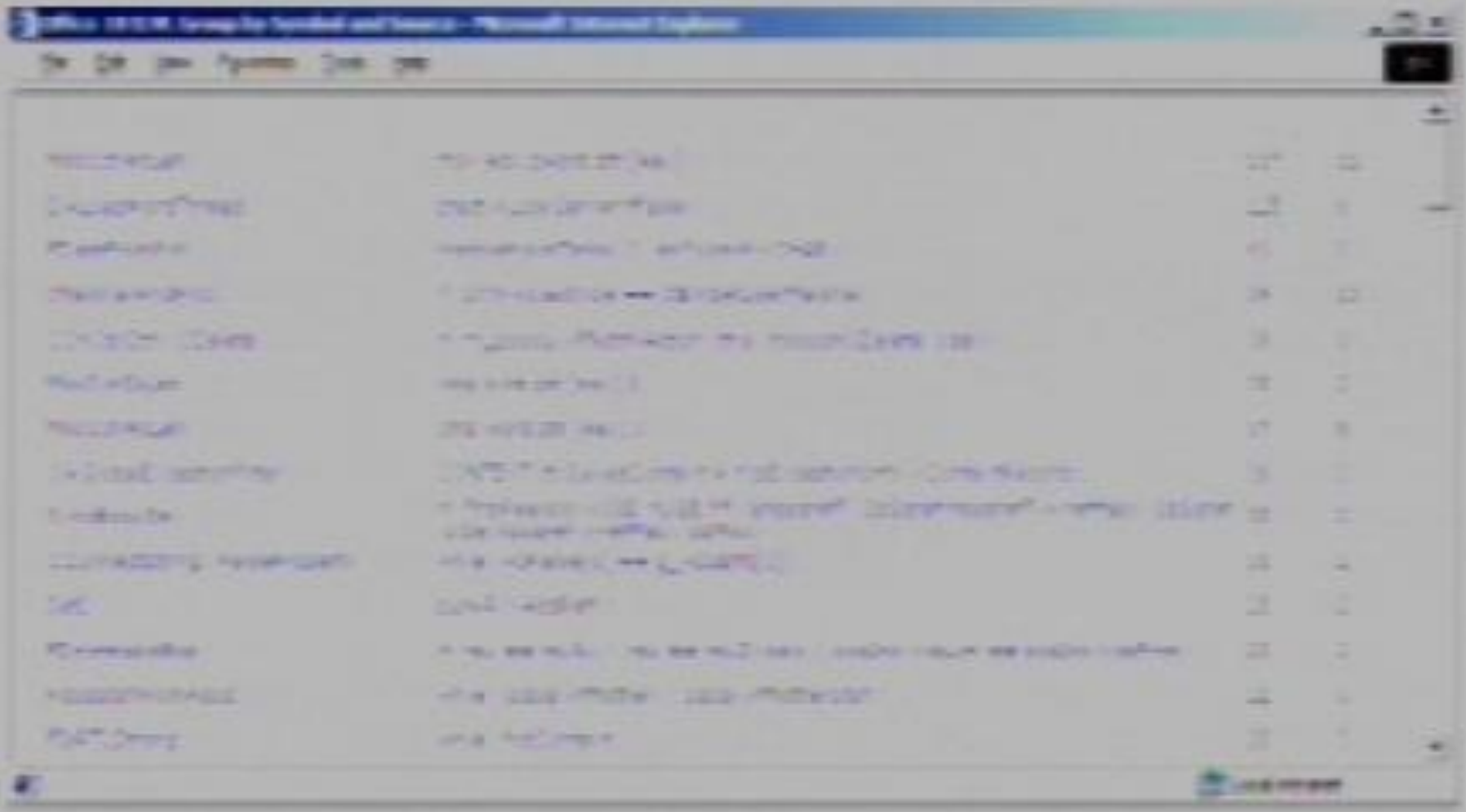


Office 10 D.W. Debug AssertTag Table

AssertTags (cont.)



OfficeWatson

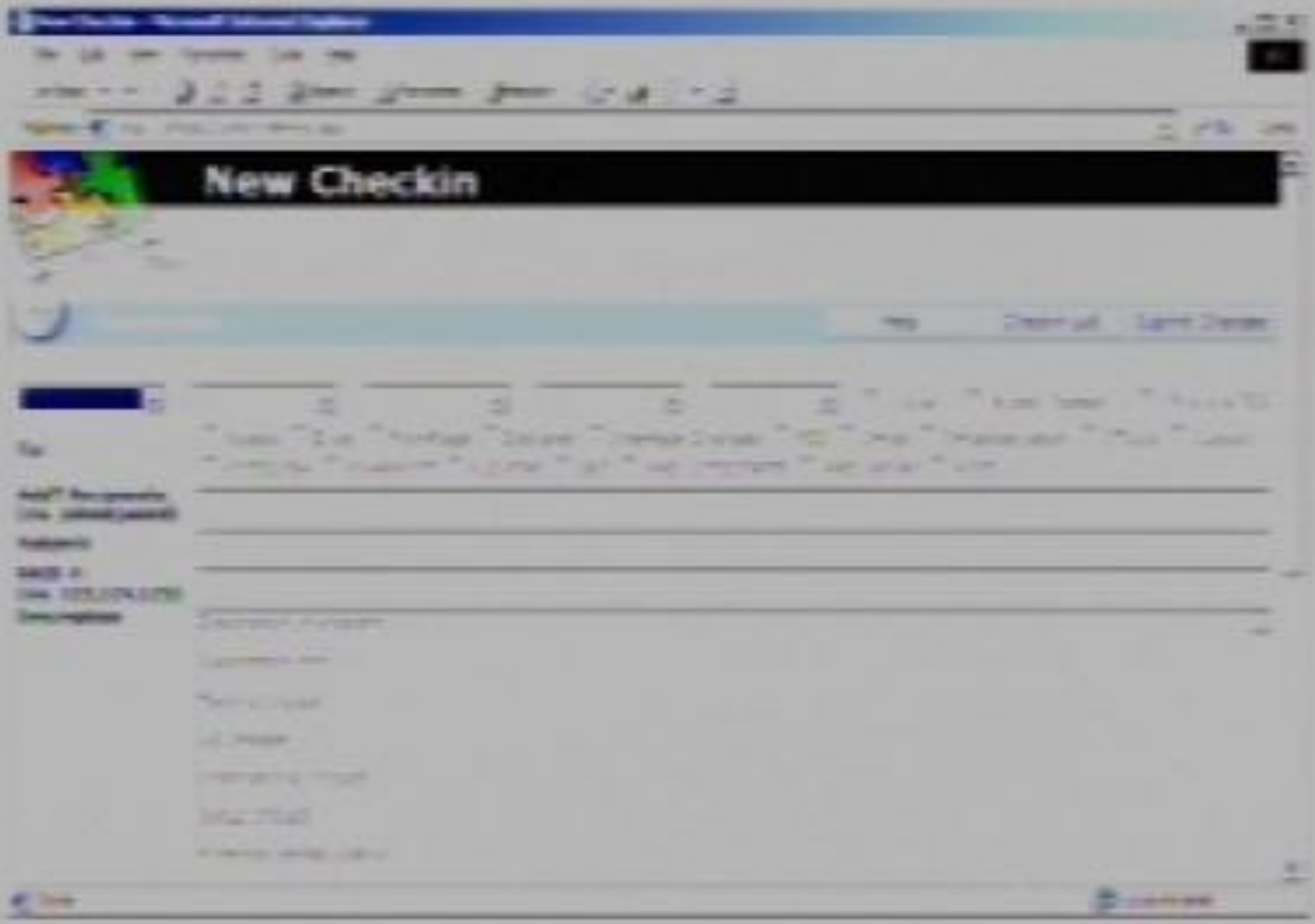


What we do well

(for developers)

- Easy to use performance tools
  - Dedicated performance team
  - Some of our own lightweight tools
- Coding/Scheduling methodologies
  - Bug age limit during milestone coding
  - Baked-in debug time on schedule
  - Checkin mail/TRD tool
    - Broadcast changes
    - TRD workflow
    - Stored in one database

Check-in tool



Things we're not good at

- Maximizing coding time
  - Coding to overhead ratio too high
  - Need more robust builds/tests
    - Too much time blocked
    - Too many "random" failures today
    - Some tools are too fragile
    - The "environment" is too fragile
  - Build break resolution/turn-around timeliness

Things we're not good at

- How do we reduce bug fixing time?
  - Improve checkin code quality
    - Verification tools
    - Better developer discipline
    - Pre-checkin test depth/targeting
    - Staging and sub-projects
  - Reduce the number of "spec issue" bugs
    - One analysis shows this is 40% of all bugs

Things we're not good at

- Reducing bug fixing time (cont.)
  - Need better way of prioritizing bugs
    - Gather more data internally & from betas
  - Performance roller coaster
    - Too much "break it now, fix it later"

Things we're not good at

- Schedule predictability
  - Better data
  - Better model of how things really are
  - Schedule *everything*